Metodología de diseño en FPGA usando Xilinx System Generator

Manuel Rodriguez Valido
Dpto. Física Fund. Exp. Electrónica y Sistemas
Universidad de La Laguna, ULL
La Laguna, Tenerife, España
mrvalido@ull.es

Martín Gutiérrez Castañeda
Dpto. Física Fund. Exp. Electrónica y Sistemas
Universidad de La Laguna, ULL
La Laguna, Tenerife, España
mgc0001@gmail.com

Eduardo Magdaleno Castello
Dpto. Física Fund. Exp. Electrónica y Sistemas
Universidad de La Laguna, ULL
La Laguna, Tenerife, España
emagcas@ull.es

David Hernández Expósito
Dpto. Física Fund. Exp. Electrónica y Sistemas
Universidad de La Laguna, ULL
La Laguna, Tenerife, España
davidhdeze@gmail.com

Fernando Pérez Nava
Dpto. Estadística, Investigación Operativa y Computación
Universidad de La Laguna, ULL
La Laguna, Tenerife, España
fdoperez@ull.es

Lucas Guerrero Vidal
Dpto. Física Fund. Exp. Electrónica y Sistemas
Universidad de La Laguna, ULL
La Laguna, Tenerife, España
lucas.guerrero@gmail.com

***Resumen*—En este trabajo se describe una metodología de diseño de sistemas digitales en FPGA mediante el uso de la herramienta Matlab-Simulink. Como ejemplo de aplicación resolvemos el algoritmo anaglifo para obtener imágenes 3D. Anaglifo es un algoritmo que a partir de imágenes estéreos se obtiene una imagen 3D que mediante gafas con filtros podemos ver la escena tridimensional. El método se aplicó con alumnos de quinto curso en ingeniería en automática y electrónica industrial.**

Keywords; XSG, IP-CORE,S FPGAs Co-simulacion, diseño digital, HDL, 3D video.

I. INTRODUCTION

El gran crecimiento tecnológico y de uso que está experimentando las FPGAs, las cuales incluyen con mas frecuencia en silicio procesadores embebidos, memorias, elementos aritméticos, interfaces de comunicaciones , etc.., posibilita la portabilidad de integrar algoritmos mas y mas complejos en ellas. Uno de los objetivos del diseño electrónico es reducir los tiempos de lanzamiento de un producto al mercado. Este hecho permite tanto reducir los costes asociados al proceso como amortizar rápidamente el proceso con los frutos de su aplicación.

Desde el punto de vista metodológico, esto se puede conseguir mediante la programación de alto nivel, que permite elevar sustancialmente la complejidad del diseño abstrayendo al diseñador de las particularidades del mismo. A esto le añadimos la posibilidad de emplear IP Cores que agilizan el procedimiento, a la vez que nos permite realizar modelos de co-diseño hardware/software para la verificación y evaluación del diseño que se plantea.

Actualmente cada fabricante de FPGA posee un entorno de desarrollo para sus dispositivos, ya sea Xilinx con el Xilinx IDE, Altera con Quartus, etc. Estos entornos pese a poseer distintas interfaces y metodologías de trabajo comparten un elemento común para la descripción de los diseños, HDL.

En el empeño de reducir el tiempo de diseño, cada vez aparecen más herramientas que permite diseñar en alto nivel. Matlab-Simulink [1,2] nos permite crear y simular el diseño mediante bloques similares a los antiguos esquemáticos. Este entorno hace uso de una librería desarrollada por Xilinx que se integra en Simulink y una herramienta denominada System Generator para DSP. Con estos dos elementos, se podrá obtener la traducción del diseño en lenguaje de más bajo nivel HDL e incluso cargar dicho diseño en una FPGA.

Los lenguajes HDL juegan un papel vital en este proceso de diseño. Los fabricantes de herramientas, en su afán de acortar el ciclo de diseño, tienen siempre como reto o sueño el siguiente límite "de la idea al Chip". Este límite está lejos, pero no es imposible de alcanzar. Matlab-Simulink, por un lado, pretende acercarse a esta idea y por otro, extender el uso de las FPGAs a entornos científicos-profesionales que actualmente hacen uso de este entorno [3,4].

El objetivo de este trabajo es presentar un metodología, para prototipado rápido de diseños digitales con FPGA, basada en un entorno de alto nivel Matlab-Simulink . Utilizaremos un enfoque basado Xilinx System Generator (XSG) y VHDL. Para mostrar la potencia de esta metodología le presentamos al alumno un diseño complejo, es decir, generar una imagen 3D (anáglifo) a partir de dos escena tomadas por dos cámaras desde dos puntos de vistas diferentes.

Hemos aplicado esta metodología en una asignatura optativa denominada Diseño Electrónico Avanzado de la titulación Ingeniería en Automática y Electrónica Industrial. Los alumnos de esta asignatura son alumnos con conocimientos en VHDL y Diseño digital en FPGA, además han usado el entorno MATLAB en algunas otras materias de la titulación. Partiendo de la experiencia previa del alumno, y en nuestra opinión, el uso de esta metodología ha sido satisfactorio ya que los alumnos de forma rápida pueden diseñar e implementar un primer prototipo de sistemas o procesador digital en FPGA.

La estructura de este trabajo es la siguiente, en primer lugar, se hace una breve descripción del entorno del trabajo así como de sus principales características, seguidamente, se detallan los pasos seguidos para elaborar el diseño, y por último se presenta la aplicación empleada para analizar el sistema así como los resultados obtenidos y conclusiones.

II. DESCRIPCIÓN DEL ENTORNO

XSG es un conjunto de bloques (librerías) integradas con MATLAB-Simulink. Dicha herramienta nos permite simular funcionalmente los diseños y usar el entorno MATLAB para verificar los modelos a nivel de bit y ciclo con los resultados de los modelos de referencias. Estos resultados de referencias pueden estar generados dentro o fuera del MATLAB. En adición a esto la herramienta nos permite programar la FPGA desde el mismo entorno proporcionado por XSG y MATLAB.

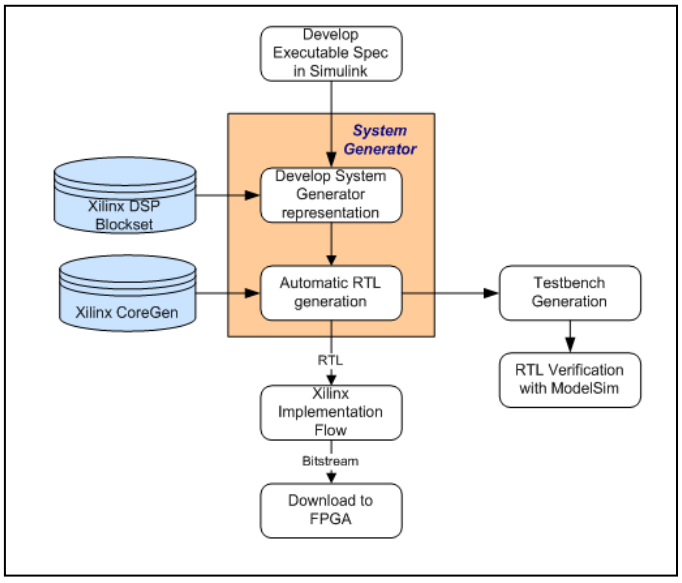


Figure 1. Flujo de diseño de System Generator

XSG complementa a las tareas convencionales de diseño con lenguajes de descripción hardware proporcionándonos un banco de pruebas fácilmente configurable tanto para la simulación funcional como la verificación hardware.

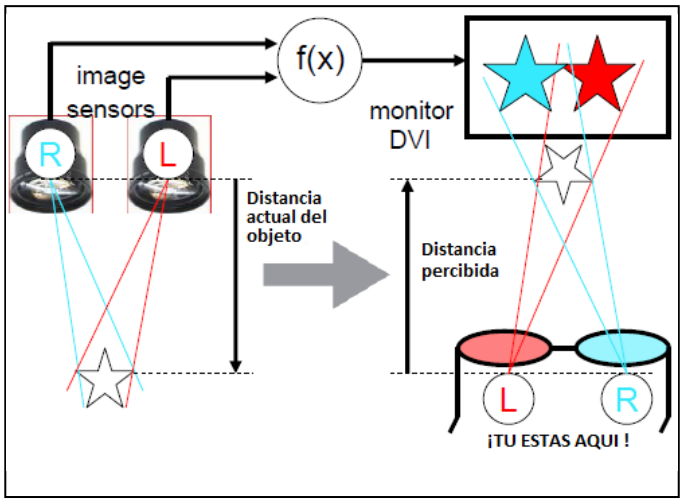


Figure 2. Sistema anaglífo. Adquisición de las dos vistas estereos y trasformación para generar la imagen 3D

La figura 1 muestra un diagrama de bloque del flujo de diseño con XSG.

Simulink proporciona un entorno gráfico para crear y modelizar un sistema. XSG consiste de una librería Xilinx blockset y Cores IP, un software para traducir los modelos Simulink a una realización hardware del modelo.

Por otro lado, XSG mapea los parámetros definidos en Simulink (mascaras y variables de las blockset) en la versión hardware del modelo (puertos, señales y atributos). En adición a esto y de forma automática XSG produce un fichero de comandos para la síntesis en la FPGA, un modelo de simulación en HDL y implementación, de tal forma no es necesario dejar el entorno grafico en ningún momento o etapa del diseño.

III. CASO A ESTUDIO

Anaglifo es una técnica que usa un par de imágenes estéreos para transmitir información de profundidad a un observador humano Fig 2. El sistema anaglifo está formado por dos sensores situados aproximadamente a 7 cm uno respecto del otro. Ambos aptan la escena y mediante una transformación matemática, la cual pondera los colores de las dos imágenes capturada genera una imagen de salida la cual tiene información ·3D de la escena percibida.

Las imágenes de anaglifo o anáglifos son imágenes de dos dimensiones capaces de provocar un efecto tridimensional, cuando se ven con lentes especiales (lentes de color diferente para cada ojo). Estas imágenes se componen de dos capas de color, superpuestas pero movidas ligeramente una respecto a la otra para producir el efecto de profundidad.

El observador, mediante las gafas anaglifo (con filtros de papel de distinto color para cada ojo) percibe la sensación de tridimensionalidad. Los filtros permiten separar el anaglifo en las dos imágenes a partir de las cuales se creó, y hace llegar cada una de ellas a cada ojo. De esta manera conseguimos que cada ojo vea la misma escena pero desde posiciones ligeramente distintas, como lo haría al observar cualquier entorno 3D, dando así sensación de profundidad [5].

Matemáticamente La técnica anaglifo viene dada por la siguiente transformación. Similar a la corrección del color de una imagen pero con dos. La salida de este proceso es una imagen obtenida partir de una combinación pesada de las imágenes capturada por los sensores izquierdos y derecho.

$$\begin{pmatrix} R_a \\ G_a \\ B_a \end{pmatrix} = \begin{pmatrix} k_{rarl} & k_{ragl} & k_{rabl} \\ k_{garl} & k_{gagl} & k_{gabl} \\ k_{barl} & k_{bagl} & k_{babl} \end{pmatrix} \times \begin{pmatrix} R_l \\ G_l \\ B_l \end{pmatrix} + \\ + \begin{pmatrix} k_{rarr} & k_{ragr} & k_{rabr} \\ k_{garr} & k_{gagr} & k_{gabr} \\ k_{barr} & k_{bagr} & k_{babr} \end{pmatrix} \times \begin{pmatrix} R_r \\ G_r \\ B_r \end{pmatrix} \quad (1)$$

El video lo podemos ver como un flujo "*streaming*" continuo de. Se caracteriza porque el flujo de datos es continuo desde el inicio de la reproducción de los contenidos hasta el fin de la misma. A menudo, en los sistemas de streaming es necesario hacer operaciones sobre el flujo de datos, esto implica que los operadores encargados de llevar a cabo estas operaciones deban hacerlas en modo en tiempo real para no interrumpir el flujo datos organizados y procedentes de una fuente. Los elementos básicos de este flujo de datos son las señales de sincronismos, los frames y los pixeles perteneciente a cada frame. Normalmente con el video suele viajar el audio.

El procesado que infiere esta expresión matricial, (1) es hecha a nivel de pixeles por cada par frame derecho e izquierdo. El procesador de streaming que implementaremos tiene que ser capaz de seguir el flujo continuo del dato. Por este motivo se basa en una arquitectura pipeline para dicha tarea.

En cuanto a las matrices de transformación *L* y *R* existen muchas variantes, para nuestro caso hemos elegido anaglifo optimizado. Las componentes del color de la imagen de salida de esa transformación esta formada como sigue:

El color rojo, R_a, por un 70% del color verde y un 30% por el azul de la imagen izquierda, el color verde G_a *y* color azul B_a es directamente el verde y azul de la imagen derecha.

Conocido los pesos de las matrices anaglifo derecha (*R*) e izquierda *(L*) y a partir video par estéreo vamos calculando en tiempo real la imagen 3D o anáglifo..

A. Metodología de diseño

La metodología de diseño con esta herramienta consiste en los siguientes pasos

1) Desarrollar el algoritmo a nivel de sistema

2) Desarrollar la implementación hardware

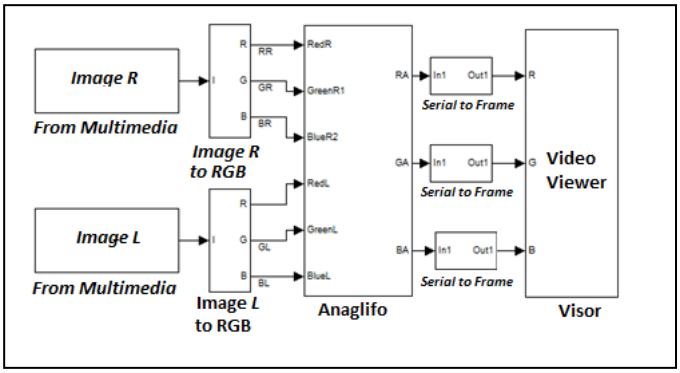
3) Valididar el algoritmo mediante hardware Co.simulation

4) Implemetar el dieño final para la FPGA.

Como fuentes de video hemos elegido unos datos generados por Visual Media Group de Microsoft Research [6]. En esta base de datos de videos 3D encontramos una secuencia de 100 imágenes tomadas por 8 cámaras y a una frecuencia de 15 frames por segundos.

Un factor importante para los diseñadores con FPGA es como mover los datos al procesador diseñado en estos dispositivos. Cada día mas los fabricantes de estos dispositivos desarrollan interfaces para facilitar la tareas de conexión de sensores con estos dispositivos. Gracias a la uso de la metodología Matlab-Simulink-XSG centramos toda nuestra atención en el desarrollo del algoritmo ya que la herramienta nos proporciona todo el soporte para ingresar y egresar los datos a las FPGA disminuyendo así el tempo de diseño y consecuentemente el "*time to marker*" . Además de esto, dicha metodología facilita y nos abstrae el uso de aritmética en punto flotante y/o en punto fija, tarea esta que puede ser muy ardua dependiendo de la complejidad del sistema. Las librerías de XSG no soportan punto flotante para ser trasladado dentro el hardware pero esta aritmética puede ser usada para la simulación a nivel de sistema y tomar los resultados como referencia y con la aritmética en punto fijo.

La figura 3 muestra un diagrama de bloques del sistema implementado. Los datos procedente de los sensores izquierdo y derecho, son generado en los bloques imagen R y imagen L . de sendas imágenes se extraen las componentes de color RGB y son introducidas en el bloque Anaglifo para aplicar el algoritmo.



.

Figure 3. Diagrama de bloques del modelo implementado en Simulink

Como comentamos anteriormente nuestra metodología se basa en 4 pasos. En el primero generamos un modelo del algoritmo a nivel de frames solo haciendo uso de las librerías de Matlab-Simulink. Este modelo funcional nos permite ver la bondad de nuestros sistemas y sus límites. Opcionalmente, podríamos usar en lugar de aritmética punto flotante aritmética

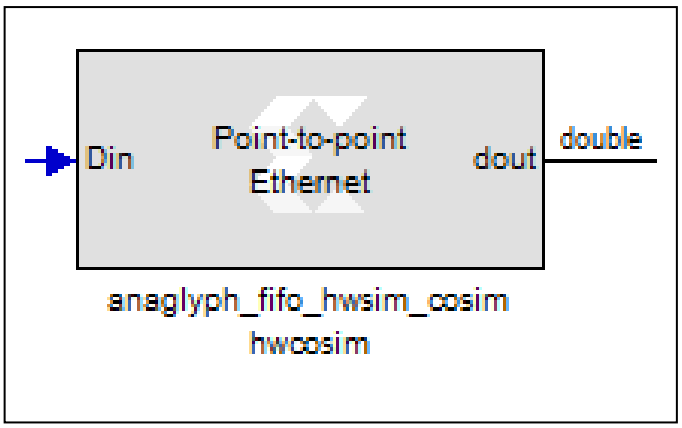


Figure 4. Componente Herdware para la co-simulación Sotfware/Hardware

en punto fijo. Esto nos permitiría generar una solución lo mas parecida posible. Una vez comprobado podría y haciendo uso de la librería Xilinx blockset realizamos la implementación hardware de nuestro sistema. En la etapa dos y tres desarrollamos la versión hardware del algoritmo el cual lo podemos comprobar al mismo tiempo que se esta ejecutando en la FPGA, Hardware co-simulacion. Una vez comprobado el modelo hardware con el teórico, se pasó a generar el componente de co-simulación (Fig. 4) y el fichero de configuración mediante la herramienta de System Generator.

El algoritmo anaglifo implementado en nuestro sistema lo podemos ver en la figura 4. En ella y a modo de diagrama de bloques podemos ver los distintos elementos de la librería blockset comentado anteriormente

IV. Resultados y Conclusiones

El resultado final de nuestro sistema puede variar según empleemos una matriz de transformación u otra. Para este ejemplo se optó por el anaglifo optimizado, con el que se obtuvieron los siguientes resultados (Fig. 5). Hemos hecho uso de una metodología de diseño basada en un entorno de alto nivel basado en MATLAB –Simulink para prototipado rápido en FPGA. Comparando esta metodología con las basadas en HDL, concluimos que: La metodología basada en Matlab-Simulink hace más intuitivo el proceso de diseño, ya que nos permite abstraernos de las particularidades hardware. Por otro lado, también se reduce el tiempo del proceso de diseño, obteniendo de forma rápida un primer modelo para poder ser implementado en la FPGA. Desde el punto de vista de simular sistemas, el hecho de hacer co-simulación hardware del modelo, reduce considerablemente el tiempo de simulación en comparación con una simulación hecha en el PC.

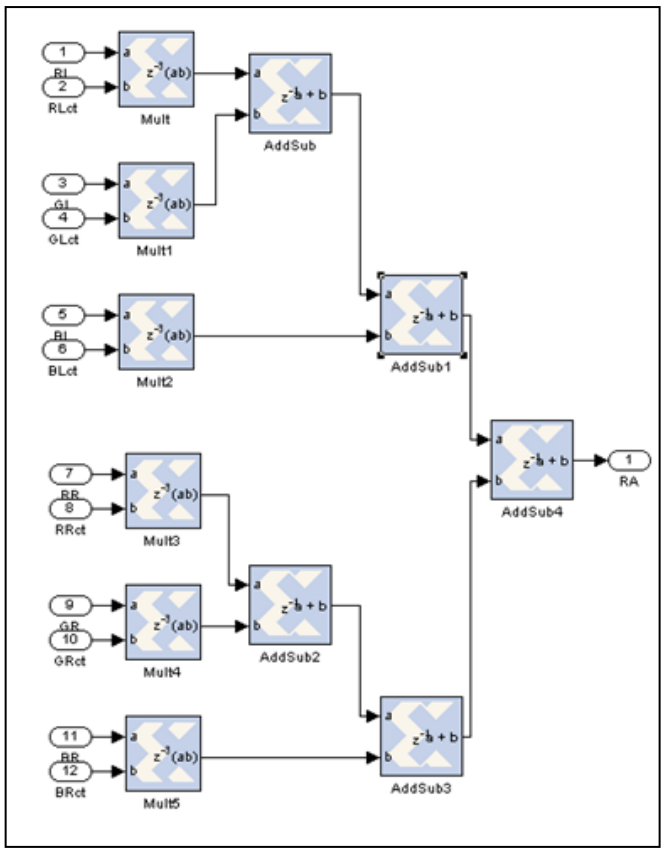


Figure 5. Algoritmo anaglifo implemntado mediante las librería Blockset de XSG

Figure 6. Imagen de salida en 3D procesada por el algoritmo Anáglifo

References.

[1] Xilinx System Generator User's Guide,2010, downloadable from;http://www. Xilinx.com.

[2] Mathworks Inc., "Simulink 3.0", http://www.mathworks.com/products/simulink/.hh

[3] Manuel Gil Rodríguez, Introducción rápida a Matlab y Simulink para ciencia e ingeniería, Ediciones Díaz de Santos, 2003

[4] Manuel Gil Rodríguez, Introducción rápida a Matlab y Simulink para ciencia e ingeniería, Ediciones Díaz de Santos, 2003.

[5] Husak, M., Guide to making your own digital stereo-video movies in DVD quality for playing on computers, 1999

[6] http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload Hhhh

[7] Rajul Dubey, Introduction to Embedded System Design Using Field Programmable Gate Arrays, Springer, 2009.